The R environment

Self-test answers

Most of the self-test answers for this chapter are provided within the chapter of the book, so they won't be included here as that would be silly.

- Create an object that represents your favourite band (unless it's Metallica, in which case use your second favourite band) and that contains the names of each band member. If you don't have a favourite band, then create an object called **friends** that contains the names of your five best friends.

One of my all-time favorite bands is Iron Maiden. If we wanted to create the object *ironMaiden* containing all the present band members' names, we would execute the following command:

```
ironMaiden<-c("Dave", "Adrian", "Nicko", "Bruce", “Janick”, “Steve”)
```

Then, if we type the command:

```
ironMaiden
```

The contents of the object *ironMaiden* will be displayed in the console window:

```
[1] "Dave" "Adrian" "Nicko" "Bruce" "Janick" "Steve"
```

- Using what you have learnt about how to use the *factor()* function, see if you can work out how to convert the **job** variable to a factor.

```
lecturerData$job<-factor(lecturerData$job, levels = c(1:2), labels = c("Lecturer", "Student"))
```

- Using the *lecturerData* dataframe, create new dataframes containing (1) the name, income and job of anyone earning 10,000 or more; (2) the name, job, income and number of friends of anyone drinking 12 units per week or less; and (3) all of the variables for those who drink 20 units or more or have a neuroticism score greater than 14.

(1)

```
highEarners <- lecturerData[income>=10000, c("name", "job", "income")]
```

Note I have used '>=' to mean 'greater than or equal to'; Alternatively, using the *subset()* function:

```
highEarners <- subset(lecturerData, income>=10000, select = c("name", "job", "income"))
```

The dataframe will be:

```
    name      job income
1    Ben Lecturer  20000
2 Martin Lecturer  40000
3   Andy Lecturer  35000
4   Paul Lecturer  22000
5 Graham Lecturer  50000
9   Mark  Student  10000
```

(2)

```
soberPeople <- lecturerData[alcohol<=12, c("name", "job", "income",  "friends")]
```

Note I have used '<=' to mean 'less than or equal to'; Alternatively, using the *subset()* function:

```
soberPeople <- subset(lecturerData, alcohol<=12, select = c("name", "job", "income",
"friends"))]
```

The dataframe will be:

```
  name      job income friends
1  Ben Lecturer  20000       5
4 Paul Lecturer  22000       4
```

(3)

```
neuroticOrAlcoholic <- lecturerData[alcohol>=20|neurotic > 14,]
```

```
neuroticOrAlcoholic <- subset(lecturerData, alcohol>=20|neurotic > 14)
```

Note I have used '|' to mean 'OR', and because we want all of the variables we haven't specified any columns. Alternatively, using the *subset()* function:

```
soberPeople <- subset(lecturerData, alcohol<=12, select = c("name", "job", "income",
"friends"))]
```

The dataframe will be:

```
    name birth_date      job friends alcohol income neurotic
2 Martin 1969-05-24 Lecturer       2      15  40000       17
3   Andy 1973-06-21 Lecturer       0      20  35000       14
5 Graham 1949-10-10 Lecturer       1      30  50000       21
6 Carina 1983-11-05  Student      10      25   5000        7
7 Karina 1987-10-08  Student      12      20    100       13
```

Oliver Twisted

Please Sir, can I have some more ... SPSS?

The following excerpt appears in:

Field, A. P. (2009). *Discovering statistics using SPSS: and sex and drugs and rock 'n' roll* (3rd edition). London: Sage.

Entering Data into the Data Editor

When you first load SPSS it will provide a blank data editor with the title *Untitled1* (this of course is daft because once it has been given the title 'untitled' it ceases to be untitled!). When inputting a new set of data, you must input your data in a logical way. The SPSS Data Editor is arranged such that *each row represents data from one entity while each column represents a variable*. There is no discrimination between independent and dependent variables: both types should be placed in a separate column. The key point is that each row represents one entity's data (be that entity a human, mouse, tulip, business, or water sample). Therefore, any information about that case should be entered across the data editor. For example, imagine you were interested in sex differences in perceptions of pain created by hot and cold stimuli. You could place some people's hands in a bucket of very cold water for a minute and ask them to rate how painful they thought the experience was on a scale of 1 to 10. You could then ask them to hold a hot potato and again measure their perception of pain. Imagine I was a participant. You would have a single row representing my data, so there would be a different column for my name, my gender, my pain perception for cold water and my pain perception for a hot potato: Andy, male, 7, 10.

The column with the information about my gender is a grouping variable: I can belong to either the group of males or the group of females, but not both. As such, this variable is a between-group variable (different people belong to different groups). Rather than representing groups with words, in SPSS we have to use numbers. This involves assigning each group a number, and then telling SPSS which number represents which group. Therefore, between-group variables are represented by a single column in which the group to which the person belonged is defined using a number. For example, we might decide that if a person is male then we give them the number 0, and if they're female we give them the number 1. We then have to tell SPSS that every time it sees a 1 in a particular column the person is a female, and every time it sees a 0 the person is a male. Variables that specify to which of several groups a person belongs can be used to split up data files (so in the pain example you could run an analysis on the male and female participants separately).

Finally, the two measures of pain are a repeated measure (all participants were subjected to hot and cold stimuli). Therefore, levels of this variable can be entered in separate columns (one for pain to a hot stimulus and one for pain to a cold stimulus).

The data editor is made up of lots of *cells*, which are just boxes in which data values can be placed. You can move around the data editor, from cell to cell, using the arrow keys ← ↑ ↓ → (found on the right of the keyboard) or by clicking the mouse on the cell that you wish to activate. To enter a number into the data editor simply move to the cell in which you want to place the data value, type the value, then press the appropriate arrow button for the direction in which you wish to move. So, to enter a row of data, move to the far left of the row, type the value and then press → (this process inputs the value and then moves you into the next cell on the right).

The first step in entering your data is to create some variables using the 'Variable View' of the data editor, and then to input your data using the 'Data View' of the data editor. We'll go through these two steps by working through an example.

The 'Variable View'

Before we input any data into the data editor, we need to create the variables. To create variables we use the 'Variable View' of the data editor. To access this view click on the 'Variable View' tab at the bottom of the data editor (Data View | **Variable View**); the contents of the window will change (see Figure 1).

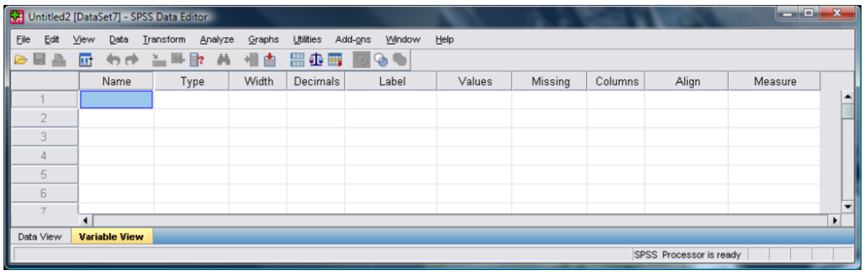

Untitled2 [DataSet7] - SPSS Data Editor
File Edit View Data Transform Analyze Graphs Utilities Add-ons Window Help
Name Type Width Decimals Label Values Missing Columns Align Measure
1 2 3 4 5 6 7
Data View Variable View
SPSS Processor is ready

Figure 1: The 'Variable View' of the SPSS Data Editor

Every row of the variable view represents a variable, and you set characteristics of a particular variable by entering information into the labelled columns. You can change various characteristics of the variable by entering information into the following columns (play around and you'll get the hang of it):

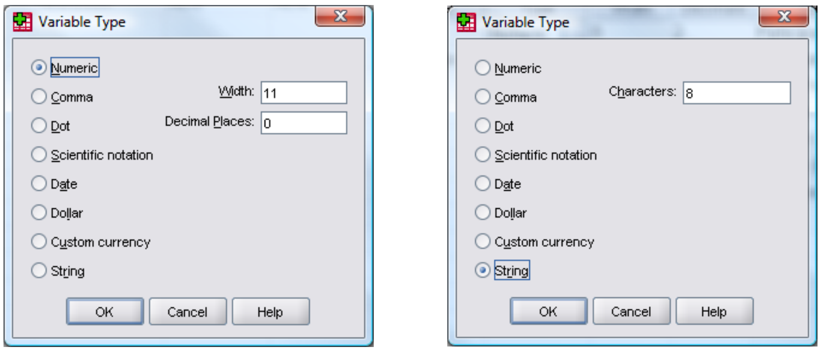
Let's use the variable view to enter the data from the book chapter (Table 3.1).

Creating a string variable

The first variable in our data set is the name of the lecturer/student. This variable consists of names; therefore, it is a *string variable*. To create this variable follow these steps:

1. Move the on-screen arrow (using the mouse) to the first white cell in the column labelled *Name*.
2. Type the word *Name*.
3. Move off this cell using the arrow keys on the keyboard (you can also just click on a different cell, but this is a very slow way of doing it).

You've just created your first variable! Notice that once you've typed a name, SPSS creates default settings for the variable (such as assuming it's numeric and assigning 2 decimal places). The problem is that although SPSS has assumed that we want a numeric variable (i.e. numbers), we don't; we want to enter people's names, namely a *string* variable. Therefore, we have to change the variable type. Move into the column labelled Type using the arrow keys on the keyboard. The cell will now look like this Numeric ... . Click on ... to activate the dialog box in Figure 2. By default, SPSS selects the numeric variable type (Numeric) — see the left panel of Figure 2. To change the variable to a string variable, click on String and the dialog box will change to look like the right panel of Figure 2. You can choose how many characters you want in your string variable (i.e. the maximum number of characters you will type for a given case of data). The default is 8, which is fine for us because our longest name is only six letters; however, if we were entering surnames as well, we would need to increase this value. When you have finished, click on OK to return to the variable view.


Variable Type

[✓ Numeric] Width: 11 Decimal Places: 0

Comma, Dot, Scientific notation, Date, Dollar, Custom currency, String

OK Cancel Help

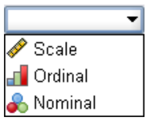
Variable Type

Numeric, Comma, Dot, Scientific notation, Date, Dollar, Custom currency, [✓ String] Characters: 8

OK Cancel Help

Figure 2: Defining a string variable in SPSS

Now because I want you to get into good habits, move to the cell in the Label column and type a description of the variable, such as 'Participant's First Name'. Finally, we can specify the level at which a variable was measured by going to the column labelled *Measure* and selecting either *Nominal*, *Ordinal* or *Scale* from the drop-down list. In this case, we have a string variable, so they represent only names of cases and provide no information about the order of cases, or the magnitude of one case compared to another. Therefore, we need to select Nominal.

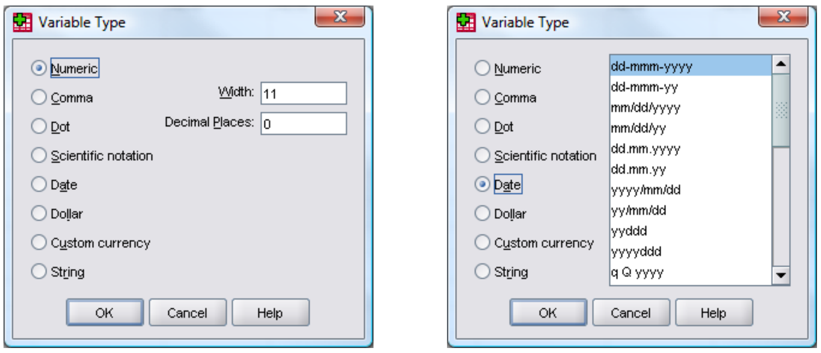


Once the variable has been created, you can return to the data view by clicking on the 'Data View' tab at the bottom of the data editor (Data View Variable View). The contents of the window will change, and you'll notice that the first column now has the label *Name*. To enter the data, click on the white cell at the top of the column labelled *Name* and type the first name, 'Leo'. To register this value in this cell, we have to move to a different cell and because we are entering data down a column, the most sensible way to do this is to press the ↓ key on the keyboard. This action moves you down to the next cell, and the word 'Leo' should appear in the cell above. Enter the next name, 'Martin', and then press ↓ to move down to the next cell, and so on.

Creating a date variable

Notice that the second column in our table contains dates (birth dates to be exact). To enter date variables into SPSS we use the same procedure as with the previous variable, except that we need to change the variable type. First, move back to the 'Variable View' using the tab at the bottom of the data editor (Data View Variable View). As with the previous variable, move to the cell in row 2 of the column labelled *Name* (under the previous variable you created). Type the word 'Birth_Date' (note that we have used a hard space to separate the words). Move into the column labelled Type using the → key on the keyboard (SPSS will create default settings in the other columns). The cell will now look like this Numeric ... . Click on ... to activate the dialog box in Figure 3. By default, SPSS selects the numeric variable type (Numeric) — see the left panel of Figure 3. To change the variable to a date, click on Date and the dialog box will change to look like the right panel of Figure 3. You can then choose your preferred date format; being British, I am used to the days coming before the month and I have stuck with the default option of dd-mmm-yyyy (i.e. 21-Jun-1973), but Americans, for example, will be used to the month and date being the other way around and could select mm/dd/yyyy

(06/21/1973). When you have selected a format for your dates, click on OK to return to the variable view. Finally, move to the cell in the column labelled *Label* and type 'Date of Birth'.



Variable Type: [✓ Numeric] Width: 11 Decimal Places: 0

Variable Type: [✓ Date] dd-mmm-yyyy

Figure 3: Defining variable types in SPSS

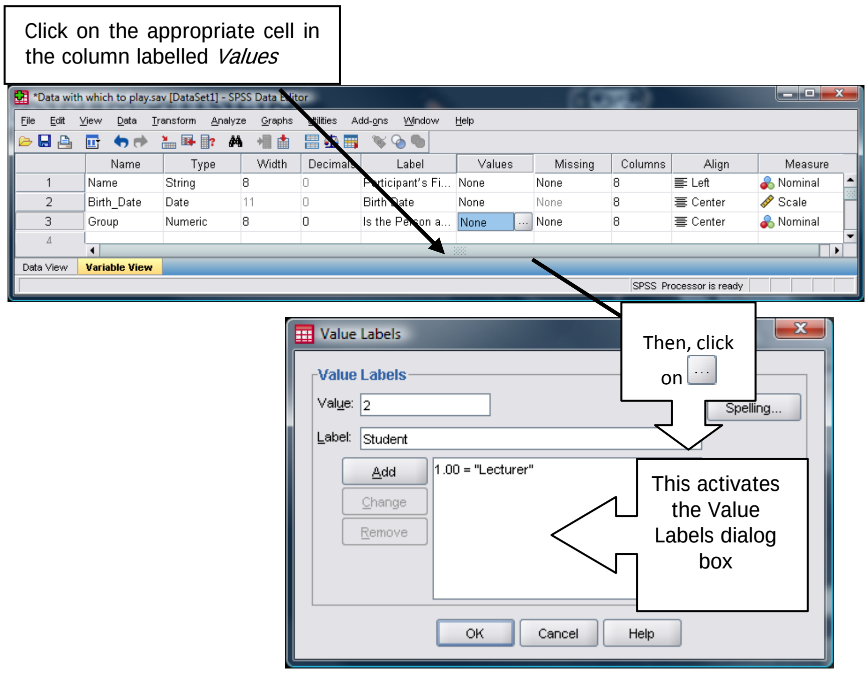
Now that the variable has been created, you can return to the data view by clicking on the 'Data View' tab (**Data View** Variable View) and input the dates of birth. The second column now has the label *Birth_Date*; click on the white cell at the top of this column and type the first value, 03-Jul-1977. To register this value in this cell, move down to the next cell by pressing the ↓ key on the keyboard. Now enter the next date, and so on.

Creating coding variables①

A coding variable (also known as a grouping variable) is a variable that uses numbers to represent different groups of data. As such, it is a *numeric variable*, but these numbers represent names (i.e. it is a nominal variable). These groups of data could be levels of a treatment variable in an experiment, different groups of people (men or women, an experimental group or a control group, ethnic groups, etc.), different geographic locations, different organizations, etc.

In experiments, coding variables represent independent variables that have been measured between groups (i.e. different participants were assigned to different groups). If you were to run an experiment with one group of participants in an experimental condition and a different group of participants in a control group, you might assign the experimental group a code of 1 and the control group a code of 0. When you come to put the data into the data editor you would create a variable (which you might call **group**) and type in the value 1 for any participants in the experimental group, and 0 for any participant in the control group. These codes tell SPSS that all of the cases that have been assigned the value 1 should be treated as belonging to the same group, and likewise for the cases assigned the value 0. In situations other than experiments, you might simply use codes to distinguish naturally occurring groups of people (e.g. you might give students a code of 1 and lecturers a code of 0).

We have a coding variable in our data: the one describing whether a person was a lecturer or student. To create this coding variable, we follow the steps for creating a normal variable, but we also have to tell SPSS which numeric codes have been assigned to which groups. So, first of all, return to the variable view (Data View **Variable View**) if you're not already in it and then move to the cell in the third row of the data editor and in the column labelled *Name* type a name (let's call it **Group**). I'm still trying to instill good habits, so move along the third row to the column called *Label* and give the variable a full description such as 'Is the person a lecturer or a student?' Then to define the group codes, move along the row to the column labelled Values and into this cell: None .... Click on ... to access the *Value Labels* dialog box (see Figure 4).



Click on the appropriate cell in the column labelled *Values*

*Data with which to play.sav [DataSet1] - SPSS Data Editor

	Name	Type	Width	Decimals	Label	Values	Missing	Columns	Align	Measure
1	Name	String	8	0	Participant's Fi...	None	None	8	Left	Nominal
2	Birth_Date	Date	11	0	Birth Date	None	None	8	Center	Scale
3	Group	Numeric	8	0	Is the Person a...	None	None	8	Center	Nominal
4										

SPSS Processor is ready

Then, click on …

Value Labels

Value: 2

Label: Student

1.00 = "Lecturer"

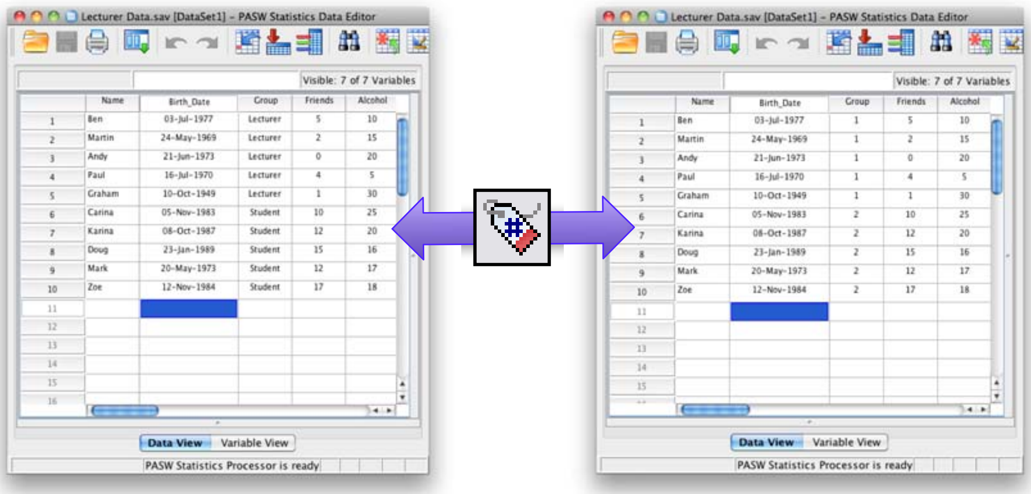
This activates the Value Labels dialog box

Figure 4: Defining coding variables and their values in SPSS

The *Value Labels* dialog box is used to specify group codes. This can be done in three easy steps. First, click with the mouse in the white space next to where it says *Value* (or press *Alt* and *u* at the same time) and type in a code (e.g. 1). These codes are completely arbitrary; for the sake of convention people typically use 0, 1, 2, 3, etc., but in practice you could have a code of 495 if you were feeling particularly arbitrary. The second step is to click the mouse in the white space below, next to where it says *Value Label* (or press *Tab*, or *Alt* and *e* at the same time) and type in an appropriate label for that group. In Figure 4 I have already defined a code of 1 for the lecturer group, and then I have typed in 2 as my code and given this a label of *Student.* The third step is to add this coding to the list by clicking on Add. When you have defined all of your coding values you can click on Spelling... and SPSS will check your variable labels for spelling errors (which can be very handy if you are as bad at spelling as I am). To finish, click on OK; if you click on OK and have forgotten to add your final coding to the list, SPSS will display a message warning you that any pending changes will be lost. In plain English this simply tells you to go back and click on Add before continuing. Finally, coding variables always represent categories and so the level at which they are measured is nominal (or ordinal if the categories have a meaningful order). Therefore, you should specify the level at which the variable was measured by going to the column labelled *Measure* and selecting Nominal (or Ordinal if the groups have a meaningful order) from the drop-down list (see earlier).

Having defined your codes, switch to the data view and type these numerical values into the appropriate column (so if a person was a lecturer, type 1, but if they were a student then type 2). You can get SPSS to

display the numeric codes, or the value labels that you assigned to them by clicking on (see


Lecturer Data.sav [DataSet1] – PASW Statistics Data Editor
Visible: 7 of 7 Variables
Data View
Variable View
PASW Statistics Processor is ready

	Name	Birth_Date	Group	Friends	Alcohol
1	Ben	03-Jul-1977	Lecturer	5	10
2	Martin	24-May-1969	Lecturer	2	15
3	Andy	21-Jun-1973	Lecturer	0	20
4	Paul	16-Jul-1970	Lecturer	4	5
5	Graham	10-Oct-1949	Lecturer	1	30
6	Carina	05-Nov-1983	Student	10	25
7	Karina	08-Oct-1987	Student	12	20
8	Doug	23-Jan-1989	Student	15	16
9	Mark	20-May-1973	Student	12	17
10	Zoe	12-Nov-1984	Student	17	18

	Name	Birth_Date	Group	Friends	Alcohol
1	Ben	03-Jul-1977	1	5	10
2	Martin	24-May-1969	1	2	15
3	Andy	21-Jun-1973	1	0	20
4	Paul	16-Jul-1970	1	4	5
5	Graham	10-Oct-1949	1	1	30
6	Carina	05-Nov-1983	2	10	25
7	Karina	08-Oct-1987	2	12	20
8	Doug	23-Jan-1989	2	15	16
9	Mark	20-May-1973	2	12	17
10	Zoe	12-Nov-1984	2	17	18

Value Labels On Value Labels Off

Figure 5), which is pretty groovy.

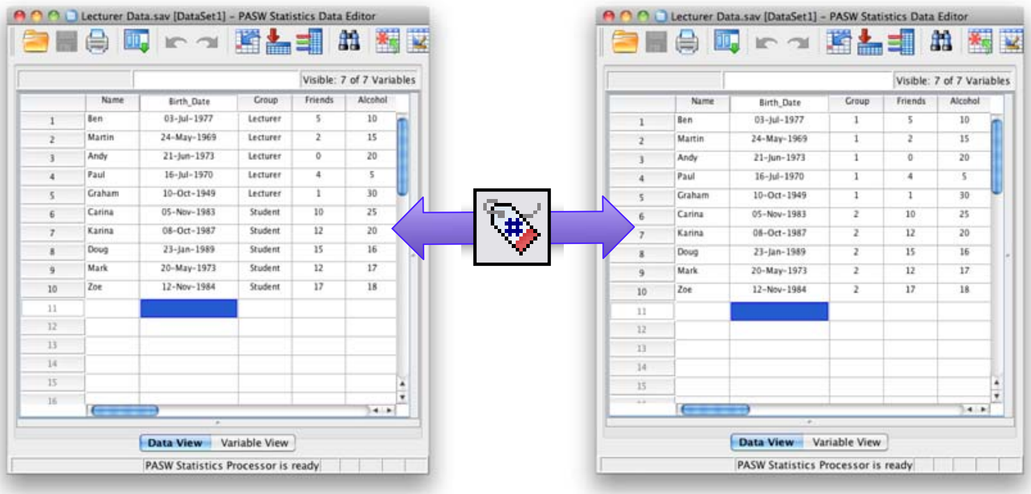
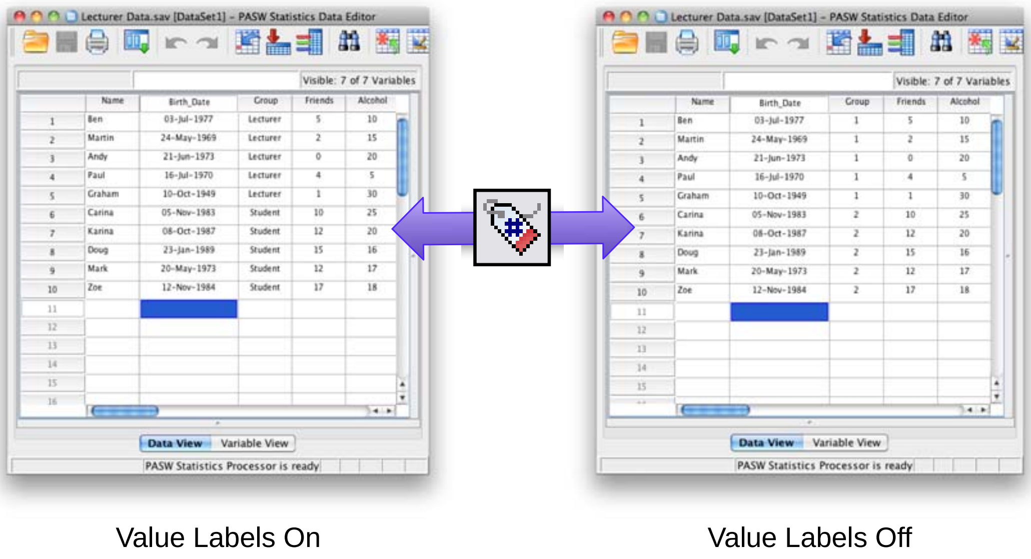

Value Labels On Value Labels Off

Figure 5 shows how the data should be arranged for a coding variable. Now remember that each row of the data editor represents data from one entity and in this example our entities were people (well, arguably in the case of the lecturers). The first five participants were lecturers whereas participants 6–10 were students.

This example should clarify why in experimental research grouping variables are used for variables that have been measured between participants: because by using a coding variable it is impossible for a participant to belong to more than one group. This situation should occur in a between-group design (i.e. a participant should not be tested in both the experimental and the control group). However, in repeated-measures designs (within subjects) each participant is tested in every condition and so we would not use this sort of coding variable (because each participant does take part in every experimental condition).



Lecturer Data.sav [DataSet1] - PASW Statistics Data Editor

Visible: 7 of 7 Variables

	Name	Birth_Date	Group	Friends	Alcohol
1	Ben	03-Jul-1977	Lecturer	5	10
2	Martin	24-May-1969	Lecturer	2	15
3	Andy	21-Jun-1973	Lecturer	0	20
4	Paul	16-Jul-1970	Lecturer	4	5
5	Graham	10-Oct-1949	Lecturer	1	30
6	Carina	05-Nov-1983	Student	10	25
7	Karina	08-Oct-1987	Student	12	20
8	Doug	23-Jan-1989	Student	15	16
9	Mark	20-May-1973	Student	12	17
10	Zoe	12-Nov-1984	Student	17	18
11					
12					
13					
14					
15					
16					

Lecturer Data.sav [DataSet1] - PASW Statistics Data Editor

Visible: 7 of 7 Variables

	Name	Birth_Date	Group	Friends	Alcohol
1	Ben	03-Jul-1977	1	5	10
2	Martin	24-May-1969	1	2	15
3	Andy	21-Jun-1973	1	0	20
4	Paul	16-Jul-1970	1	4	5
5	Graham	10-Oct-1949	1	1	30
6	Carina	05-Nov-1983	2	10	25
7	Karina	08-Oct-1987	2	12	20
8	Doug	23-Jan-1989	2	15	16
9	Mark	20-May-1973	2	12	17
10	Zoe	12-Nov-1984	2	17	18
11					
12					
13					
14					
15					

Figure 5: Coding values in the data editor with the value labels switched off and on

Creating a numeric variable

Numeric variables are the easiest ones to create because SPSS assumes this format for data. Our next variable is **No. of friends**; to create this variable we move back to the variable view using the tab at the bottom of the data editor (Data View **Variable View**). As with the previous variables, move to the cell in row 4 of the column labelled *Name* (under the previous variable you created). Type the word 'Friends'. Move into the column labelled Type using the → key on the keyboard. As with the previous variables we have created, SPSS has assumed that this is a numeric variable, so the cell will look like this Numeric .... We can leave this as it is, because we do have a numeric variable.

Notice that our data for the number of friends has no decimal places (unless you are a very strange person indeed, you can't have 0.23 of a friend). Move to the Decimals column and type '0' (or decrease the value from 2 to 0 using ⇳) to tell SPSS that you don't want any decimal places.

Next, let's continue our good habit of naming variables and move to the cell in the column labelled *Label* and type 'Number of Friends'. Finally, we can specify the level at which a variable was measured by going to the column labelled *Measure* and selecting Scale from the drop-down list (this will have been done automatically actually, but it's worth checking).

Once the variable has been created, you can return to the data view by clicking on the 'Data View' tab at the bottom of the data editor (**Data View** Variable View). The contents of the window will change, and you'll notice that the first column now has the label *Friends*. To enter the data, click on the white cell at the top of the column labelled *Friends* and type the first value, 5. To register this value in this cell, we have to move to a different cell and because we are entering data down a column, the most sensible way to do this is to press the ↓ key on the keyboard. This action moves you down to the next cell, and the number 5 should appear in the cell above. Enter the next number, 2, and then press ↓ to move down to the next cell, and so on.

Please Sir, can I have some more ... data restructuring?

To restructure the *satisfactionData* dataframe we create a new dataframe (which I have unimaginatively called *restructuredData*). This dataframe is based on the existing data (*satisfactionData*), but we use *reshape()* to, as the name suggests, reshape it. This function takes the general form:

```
newDataFrame<-reshape(oldDataFrame, idvar = c(constant variables),
varying = c(variables that change across columns), v.names = "Name of
Variable to contain Scores", timevar = "Name of Index Variable", times =
c(numbers representing levels of the index variable), direction = "long")
```

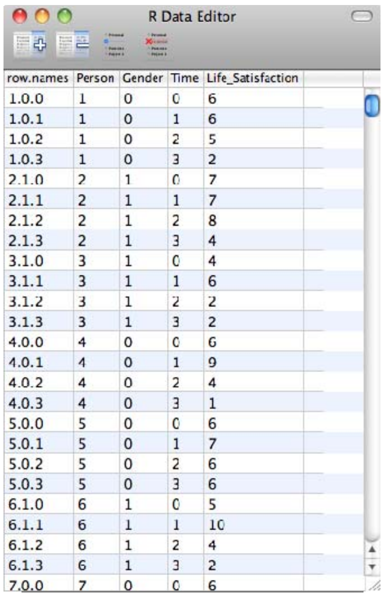
As you can see, the *reshape()* command has a lot of options contained within it and we will have a look at these in turn:

- **idvar**: This option specifies any variables in the dataframe that do not vary over time. For these data we have two variables that don't vary over time, the first is the person's identifier (**Person**), and the second is their gender (**Gender**). We can specify these variables as *idvar = c("Person", "Gender")*.
- **varying**: This option specifies the variables that do vary over time. In other words, it specifies the names of variables currently in different columns that you would like to be restructured so that they are in different rows. We have four columns that we want to restructure (**Satisfaction_Base**, **Satisfaction_6_Months**, **Satisfaction_12_Months**, **Satisfaction_18_Months**). These can be specified as *varying = c("Satisfaction_Base", "Satisfaction_6_Months", "Satisfaction_12_Months", "Satisfaction_18_Months")*.
- **v.names**: This option allows you to specify a name for the outcome variable. Our scores represent life satisfaction so we could set the variable name accordingly using *v.names = "Life_Satisfaction"* (note that because the variable name is text it needs to be enclosed in quotation marks).
- **timevar**: This option allows you to specify a name for the index variable (described above). By default, it is assumed that your columns represent different points in time and the new variable is, therefore, called 'time'. However, your columns might have represented different conditions in a repeated measures experiment, in which case you will want to call this variable something else. Although for these data the default name of 'time' is fine, I have set the variable name using *timevar = "Time"* to give you an idea of how you'd change the name, and also because I'm a pedant and wanted a capital 'T' at the start of 'time'. I need to get a life.
- **times**: This option enables you to specify the values used by the index variable to denote different levels. For example, we had four columns that we have restructured, so we have four levels of our index variable. By default, **R** will simply code these as 1, 2, 3, 4. Seems logical enough, but you might want to change this default. For this example, it's useful to centre this variable at 0 because our initial life satisfaction was measured before the new relationship. Therefore, a baseline of 0 is meaningful for these data because it is the value of life satisfaction when not in a relationship. Therefore, we want to use index values of 0, 1, 2, 3 rather than 1, 2, 3, 4. This can be achieved by using *times = c(0:3)*. This command just tells **R** to use the values 0 to 3 inclusive as the index values. If you don't want a sequence of numbers then specify the values individually (e.g. *times = c(-1, 2, 5, 10)*).
- **direction**: This option can be set to either *"wide"* or *"long"* and determines the direction in which you want to reshape the data. In this case we want to create a long format data file so we would specify *direction = "long"*.

If we piece all of these options together, we get the following command:

```
restructuredData<-reshape(satisfactionData, idvar = c("Person", "Gender"), varying =
c("Satisfaction_Base", "Satisfaction_6_Months", "Satisfaction_12_Months",
"Satisfaction_18_Months"), v.names = "Life_Satisfaction", timevar = "Time", times =
c(0:3), direction = "long")
```

If you execute this command, you should find that your data has been restructured to look like this:


R Data Editor

row.names	Person	Gender	Time	Life_Satisfaction
1.0.0	1	0	0	6
1.0.1	1	0	1	6
1.0.2	1	0	2	5
1.0.3	1	0	3	2
2.1.0	2	1	0	7
2.1.1	2	1	1	7
2.1.2	2	1	2	8
2.1.3	2	1	3	4
3.1.0	3	1	0	4
3.1.1	3	1	1	6
3.1.2	3	1	2	2
3.1.3	3	1	3	2
4.0.0	4	0	0	6
4.0.1	4	0	1	9
4.0.2	4	0	2	4
4.0.3	4	0	3	1
5.0.0	5	0	0	6
5.0.1	5	0	1	7
5.0.2	5	0	2	6
5.0.3	5	0	3	6
6.1.0	6	1	0	5
6.1.1	6	1	1	10
6.1.2	6	1	2	4
6.1.3	6	1	3	2
7.0.0	7	0	0	6

Smart Alex's solutions

Task 1

- Smart Alex's first task for this chapter is to save the data that you've entered in this chapter. Save it somewhere on the hard drive of your computer (or a USB stick if you're not working on your own computer). Give it a sensible title and save it somewhere easy to find (perhaps create a folder called 'My Data Files' where you can save all of your files when working through this book).

The data that we need to save are the **Lecturer Data** that you should have entered in this chapter. First of all, let's create a folder on the hard drive of your computer called 'My Data Files'. To do this you could go to your 'Documents' folder and then create a new folder by using the **File⇒New Folder** menu, and then type in a name for your new folder such as 'My Data Files'. So, assuming your user name is Andy F, the new folder would have the following file path:

```
C:/Users/Andy F/Documents/My Data Files/
```

Before we can save the Lecturer Data in this folder, we first need to set this folder as our working directory so that **R** knows this is the location that we want to save the data. To set the working directory to be this folder, we use the *setwd()* command to specify the newly created folder as the working directory:

```
setwd("C:/Users/Andy F/Documents/My Data Files")
```

By executing this command, we can now save and access files in that folder directly.

Earlier on we created a dataframe called *lecturerData.* To export (save) this dataframe to a tab-delimited text file called **Lecturer Data.txt**, we could execute this command:

```
write.table(lecturerData, "Lecturer Data.txt", sep="\t", row.names = FALSE)
```

Or we could save the dataframe as a comma-separated values file by executing the following command:

```
write.csv(lecturerData, "Lecturer Data.csv")
```

Task 2

- Your second task is to enter the data that I used to create Figure 3.8. These data show the score (out of 20) for 20 different students, some of whom are male and some female, and some of whom were taught using positive reinforcement (being nice) and others who were taught using punishment (electric shock). Just to make it hard, the data should not be entered in the same way that they are laid out below:

Male		Female	
Electric Shock	Being Nice	Electric Shock	Being Nice
15	10	6	12
14	9	7	10
20	8	5	7
13	6	4	8
13	7	8	13

The first thing to note is that we have a coding variable in our data: the one describing whether a student was taught using punishment (electric shock) or reinforcement (being nice). To create this coding variable, we follow the steps for creating a normal variable, but we also have to tell **R** that the variable is a coding variable/factor and which numeric codes have been assigned to which groups.

First off, we can enter the data and then worry about turning these data into a coding variable. In our data we have 10 students who were taught using the electric shock method (whom we will code with 1) and 10 students who were taught using the method of being nice (whom we will code with 2). As such, we need to enter a series of 1s and 2s into our new variable, which we'll call **Method**, we could do this using the *rep()* function.

```
Method<-c(rep(1, 10), rep(2, 10))
```

To turn this variable into a factor, we use the *factor()* function:

```
Method<-factor(Method, levels = c(1:2), labels = c("Electric Shock", "Being Nice"))
```

Gender is also a coding variable. There are 10 males (whom we will code with 0) and 10 females (whom we will code with 1). As such, we need to enter a series of 0s and 1s into our new variable, which we'll call **Gender**, we could do this using the *rep()* function:

```
Gender<-c(rep(0, 5),rep(1, 5), rep(0, 5),rep(1, 5))
```

As you will see, I have put *rep(0, 5),rep(1, 5)* in twice, rather than *rep(0, 10),rep(1, 10)*. This is because 5 males and 5 females were taught using punishment and 5 males and 5 females were taught using reinforcement, and so we need to tell **R** to list the first 5 males and the first 5 females, followed by the second 5 males and the second 5 females (not all 10 males followed by all 10 females) so that each gender correctly matches up with the corresponding teaching method. To turn this variable into a factor, we use the *factor()* function:

```
Gender<-factor(Gender, levels = c(0:1), labels = c("Male", "Female"))
```

Our next variable, which I have called **Mark**, is a numeric variable. We need to enter the data into the command in the correct order so that **R** will know which mark belongs to whom. They need to be in the order: Male 'Electric Shock', Female 'Electric Shock', Male 'Being Nice' and Female 'Being Nice' as in the following command:

```
Mark<-c(15,14,20,13,13,6,7,5,4,8,10,9,8,6,7,12,10,7,8,13)
```

Having created the individual variables we can bind these together in a dataframe. We do this by executing the following command:

```
teachingMethodData<-data.frame(TeachingMethod, Gender, Mark)
```

You can then view the data by typing the command:

```
teachingMethodData
```

The data can be found in the file **teachingMethodData.txt** and should look like this:

```
        Method      Gender   Mark
```

```
1  Electric Shock   Male   15
2  Electric Shock   Male   14
3  Electric Shock   Male   20
4  Electric Shock   Male   13
5  Electric Shock   Male   13
6  Electric Shock Female    6
7  Electric Shock Female    7
8  Electric Shock Female    5
9  Electric Shock Female    4
10 Electric Shock Female    8
11     Being Nice   Male   10
12     Being Nice   Male    9
13     Being Nice   Male    8
14     Being Nice   Male    8
15     Being Nice   Male    7
16     Being Nice Female   12
17     Being Nice Female   10
18     Being Nice Female    7
19     Being Nice Female    8
20     Being Nice Female   13
```

To save the data as a tab-delimited file (or you could save it as a CSV file if you prefer), we use the *write.table()* command:

```
write.table(teachingMethodData, "teachingMethodData.txt", sep="\t", row.names=FALSE)
```

Remember:

1) The data will be saved in the file that you previously set as your working directory (assuming you have set one, otherwise who knows where it will save it!).

2) The above command will save the data only, *not* the commands you used to enter the data. If you want to save the commands for future use, the best method it is to type the commands into the editor window rather than the **R** console (you can copy and paste commands from the **R** console to the editor window if you have used the **R** console), and then save the editor window in a sensible place on your hard drive, ideally in the folder that you use as your working directory.

Task 3

- Research has looked at emotional reactions to infidelity and found that men get homicidal and suicidal and women feel undesirable and insecure (Shackelford, LeBlanc, & Drass, 2000). Let's imagine we did some similar research: we took some men and women and got their partners to tell them they had slept with someone else. We then took each person to two shooting galleries and each time gave them a gun and 100 bullets. In one gallery was a human-shaped target with a picture of their own face on it, and in the other was a target with their partner's face on it. They were left alone with each target for 5 minutes and the number of bullets used was measured. The data are below; enter them into **R** and save them as **Infidelity.csv** (clue: they are not entered in the format in the table!).

Male		Female	
Partner's Face	Own Face	Partner's Face	Own Face
69	33	70	97
76	26	74	80
70	10	64	88
76	51	43	100
72	34	51	100
65	28	93	58
82	27	48	95
71	9	51	83
71	33	74	97
75	11	73	89
52	14	41	69
34	46	84	82

This experiment is a within-subjects design, all participants (male and female) participated in both experimental conditions (bullets shot at own face and bullets shot at partner's face). However, there is one coding variable, which is whether the person aiming the bullets was male or female. There are

12 males (whom we will code with 0) and 12 females (whom we will code with 1). As in Task 2, we need to enter a series of 0s and 1s into our new **Gender** variable and we can do this using the *rep()* function:

```
Gender<-c(rep(0, 12), rep(1, 12))
```

We then use the *factor()* function to turn this variable into a factor:

```
Gender<-factor(Gender, levels = c(0:1), labels = c("Male", "Female"))
```

The next variable, which we will call **Partner,** refers to the number of bullets each person shot at their partner's face. This is a numeric variable and can be entered using the following command:

```
Partner<-c(69,76,70,76,72,65,82,71,71,75,52,34,70,74,64,43,51,93,48,51,74,73,41,84)
```

The final variable is also a numeric variable and refers to the number of bullets each person shot at their own face. We could call this variable **Self** and create it by executing the following command:

```
Self<-c(33,26,10,51,34,28,27,9,33,11,14,46,97,80,88,100,100,58,95,83,97,89,69,82)
```

Having created the individual variables we can bind these together in a dataframe. We do this by executing the following command:

```
infidelityData<-data.frame(Gender, Partner, Self)
```

We can then view the data by executing:

```
infidelityData
```

The data can be found in the file **Infidelity.csv** and should look like this:

```
   Gender  Partner Self
1    Male       69   33
2    Male       76   26
3    Male       70   10
4    Male       76   51
5    Male       72   34
6    Male       65   28
7    Male       82   27
8    Male       71    9
9    Male       71   33
10   Male       75   11
11   Male       52   14
12   Male       34   46
13 Female       70   97
14 Female       74   80
15 Female       64   88
16 Female       43  100
17 Female       51  100
18 Female       93   58
19 Female       48   95
20 Female       51   83
21 Female       74   97
22 Female       73   89
23 Female       41   69
24 Female       84   82
```

To save these data as a CSV file we can use the *write.csv()* command:

```
write.csv(infidelityData, "Infidelity Data.csv")
```